FLUX FILE PRIMER

OR: HOW I LEARNED TO STOP WORRYING AND LOVE THE FLUX

Contents

Overview

The simulation of neutrino experiments is significantly different from other HEP physics experiments (both fixed target and collider) because of the complications introduced by the complex means used to generate a source of neutrinos. The spectrum of neutrinos is not mono-energetic, nor well collimated (which has effects that need to be accounted for in any near detector).

Major Steps in the Simulation Chain:

- The source of protons (NuMI or Booster) is simulated [this is generally trivial]
- The physics of protons and their secondaries impacting the target and beamline is simulated. The decays of hadrons and muons that give rise to one or more neutrinos produced by the beamline is recorded in files commonly, but misleadingly, referred to as "flux files"[1]. These decays are normally sampled to suppress some of the low energy decays (with remaining entries tagged with an accompanying importance weight) to save space.
- To be used by the detector simulation, these "dk2nu" files must be interpreted to produce individual neutrino "rays" in the reference frame of the particular detector. As well as dealing with the importance weight, this involves a calculation arising from relativity that gives the neutrino's energy and weight for forcing the decay to go through a particular point[2].
- The results of this de-weighting process can either directly be embedded in the event generator, or handled as a separate step and the output written to a simplified format file[3]. Such files, once made, make further steps more efficient, but are relevant for only one detector location. The trade-off on making this separate might depend on which detector is being simulated and as well as concerns about information loss in the intermediate step.
- A representation of the detector geometry is exposed to neutrino "rays" and the physics of neutrino interactions is simulated which give rise to particles leaving a struck nucleus. This is referred to as the "event generator" stage (e.g. GENIE).
- Finally a simulation of the physics of particle propagation in the detector (e.g. Geant4) and the detector's response is made, yielding the equivalent of raw data from a real DAQ system. Some of these step may or may not be combined with event generator step.

---

[1] In this document we will refer to these as "dk2nu" files which is the up-and-coming format for these files; older variants might refer to them as "g4numi", "g4mnv" (Minerva), "g4lbne" or "flugg" files. Even there there are incompatible variants.

[2] For the GENIE neutrino event generator the code that does this is an instance of a "flux driver".

[3] One common type of resulting de-weighted files are called GSimpleNtpFlux files, or "gsimple" files. GENIE code that reads this format of files is a different instance of a "flux driver".

Additional Complications:

Combining the simulation of the beamline and the detector would be computationally impractical, so the output of the beam simulation must be written out in some form. Besides the two flavors of files (e.g. "dk2nu" and "gsimple") additional combinatorics arise from variations on:

- Changes in the beamline geometry (e.g. target construction, relative placement of beamline elements) and parameters that control the magnetic properties (e.g. horn current). These can arise from differences in design vs. built, studies to investigate same, or simply standard changes in operating parameters.
- Fundamental differences in the underlying beamline simulation. There are currently two competing frameworks in use: FLUGG and Geant4. Within Geant4 there are different options for how this physics is simulated ("physics lists") that affect the fidelity of the simulation.
- For use by the on-axis detectors one can make additional cuts during "dk2nu" generation that suppress decays that result in neutrinos that are not detectable by the detector, but off-axis detectors need to keep those decays ("lowth" files).

Unfortunately all attempts to impose a standardized nomenclature for referring to different features have been subject to historical vagaries, subverted or partially ignored, so external references (or even some herein) might be inconsistent.

Primary Results From The Beamline Simulation

The "dk2nu" files contain the information about neutrinos that arise from decays of hadrons and muons in the beamline. The details of what is store is documented elsewhere (link goes here). They hold sufficient information about the kinematics of their ancestry that they can be re-weighted for different locations. The new "dk2nu" format significantly expands upon, as well as unifying the naming conventions of, most of the previous formats (e.g. the basic "flugg" and "g4numi" formats). Scripts exist to convert the older formats to the "dk2nu" files, though that process necessarily can't fill in the pieces that weren't record. Newer versions of beamline simulations will write this format directly. It follows that such files will be larger than older files.

Generating "gsimple" Files

The process of de-weighting "dk2nu" (or equivalent) files to individual neutrino rays can be computationally expensive, especially for off-axis detectors (i.e. NOvA, or uBooNE viewing the NuMI beam). Scripts exist to convert files of decays to "gsimple" format (link goes here).

The size and number of "gsimple" files generated from a set of "dk2nu" files is tunable when they are being generated. Each file represents the neutrinos rays traveling through a "flux window" for a given number of protons-on-target (POTs). The size of the flux window might be chosen differently even for the same detector location depending on the desire to use the file to generate events in the rock surrounding the detector proper.

File Handling in ART/GENIEHelper

The majority of the FNAL neutrino experiments (though notably not Minerva or MINOS) are using the ART framework for their offline code. Each of these makes use of a shared "nusoft" code (GENIEHelper) to interface the GENIE event generator to the ART framework. The liquid argon (LArSoft) and NOvA experiments use different ART modules to format the results into their output files, but the basics of configuring GENIE are shared.

One aspect of this GENIEHelper code is how it handles flux files; in GENIEHelper this is common to both the handling of "dk2nu" and "gsimple" files. The way the code currently works (subject to revision) is to be given a pattern, and implicitly a set of top-level directory paths (stems). The code prepends each of the stems to the pattern, uses the standard C library code 'glob' to resolve the wildcards into a list of files, and accumulates a list of all potential files.

The GENIEHelper code then (optionally) randomizes the list of potential files, which is useful for adding entropy to the simulation. It then walks down the list of files using ROOT's gSystem->GetPathInfo to determine file size (which for normal files uses 'stat'), accumulating a list of files to actually use until it reaches a user defined limit (default 2000MB).

Note the use of 'glob' here currently precludes the use of alternative protocols that do not look like ordinary files. This is generally fine because the code will run through the entries in the files sequentially fast enough and often more than once, so that it would be detrimental to get this data on the fly from a non-local source. Network reads of the data would be very inefficient, with additional latency as reading comes in small chunks for each request of an entry and the potential for re-reading entries when each pass through the files exhausts the data.

Future alternatives might involve:

1. Handle non-direct file protocols simply by bypassing all this code and taking the list of files as is. This would require tacking onto the .fcl file the explicit list of files for each job. It is presupposed (but not verified) that all the code could handle non-direct file protocols if the glob/stat bits were bypassed. This moves the selection logic out of GENIEHelper into something that appends to the .fcl file. The .fcl files parameter is an array, so the entries *don't* have to be wildcarded, one can list them all explicitly.

2) Write some protocol aware alternative to the glob/stat issue (GetPathInfo to determine file size might work as is (needs to be checked) -- but 'glob' is undoubtably doesn't) into GENIEHelper.

The first (where one makes the list external to the ART job itself and manipulates a base .fcl file) was deemed "undesirable" at the time because it adds complication to both production and user initiated jobs; but that decision could be revisited by ThePowersThatBe.

Representative File Samples

Currently the most interesting file samples are those that represent the NOvA-era target in the nominal position with the nominal (forward, i.e. +200) horn current. But other configurations produce different size files for the same number protons-on-target (POTs). Normal flugg/g4numi files represent 500,000 POTs.

set	target	pull-back	horn current	cuts	format	# files	total size (MB)
n1	NOvA (mn)	0	200	lowth	flugg	968	160771
n2	NOvA (mn)	0	-200	lowth	flugg	968	156881
n3	MINOS (le)	10	185	lowth	flugg	200	78290
	NOvA (me)	0	200	?	g4mnv	500	96049
	NOVA (me)	0	-200	?	g4mnv	500	88012
	NOvA (mn)	0	200	nova-fd	gsimple	100	14277
	NOvA (mn)	0	-200	nova-fd	gsimple	100	12013
	NOvA (mn)	0	200	nova-nd	gsimple	200	37104
	NOvA (mn)	0	-200	nova-nd	gsimple	100	30692
	NOvA (mn)	0	200	nova-ndrock	gsimple	100	17101
	NOvA (mn)	0	-200	nova-ndrock	gsimple	100	13960

Bibliography

Author Last Name, First Name. “Book Title or Reference Title.” City: Publisher, Date.

blah

Representation of times and dates http://en.wikipedia.org/wiki/ISO_8601